Soluções para problemas difíceis
Algoritmos II

Filipe Pirola Santos - 2021031580

[1]Departamento de Computação – Universidade Federal de Minas Gerais (UFMG)

filipepirolasantos@hotmail.com

Resumo. *Este trabalho propõe o desenvolvimento e a análise comparativa de soluções para o problema do Caixeiro Viajante (TSP) por meio de três algoritmos distintos: Branch and Bound, Christofides e Twice Around the Tree. O objetivo é investigar e comparar as implementações e eficácias de cada abordagem na resolução deste problema.*

1. Introdução

O Problema do Caixeiro Viajante (TSP) representa um desafio clássico na área de otimização combinatória, onde o objetivo é encontrar o menor caminho que visita cada cidade exatamente uma vez e retorna ao ponto de origem. Diante da relevância desse problema em diversas aplicações práticas, desde logística até planejamento de rotas, a busca por algoritmos eficientes para sua resolução é incessante.

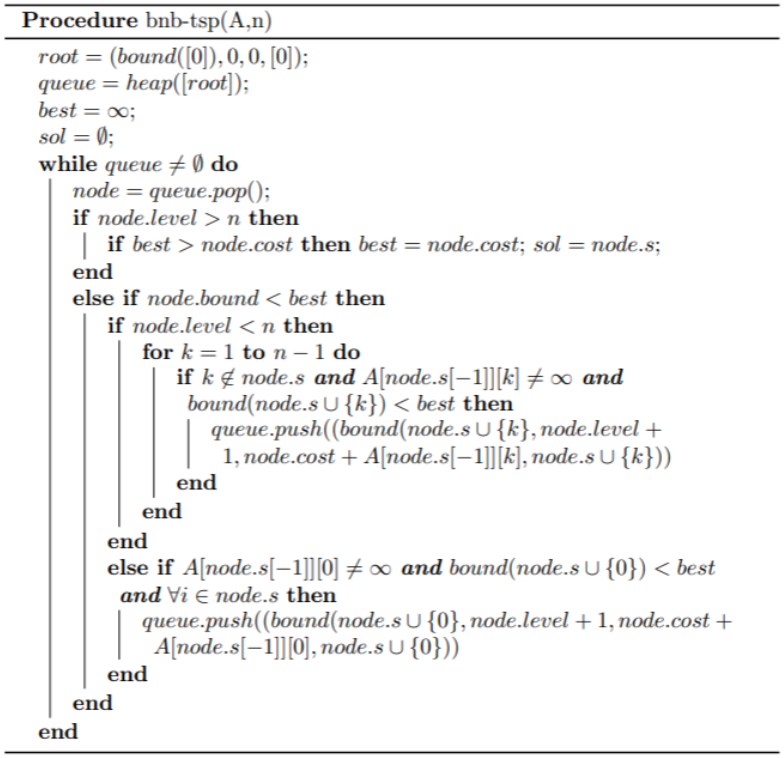
Este trabalho tem como foco o desenvolvimento e a análise comparativa de soluções para o TSP por meio de três algoritmos distintos: Branch and Bound, Christofides e Twice Around the Tree. Cada abordagem representa uma estratégia única na resolução desse desafio, abrangendo desde métodos exatos até heurísticas que visam encontrar soluções próximas à ótima em tempo hábil.

O algoritmo Branch and Bound destaca-se pela sua capacidade de explorar sistematicamente o espaço de soluções, buscando eficiência na obtenção da solução ótima. Já o algoritmo de Christofides, conhecido por sua aproximação polinomial, oferece uma alternativa rápida e eficaz para instâncias do TSP. Por último, o Twice Around the Tree, um algoritmo heurístico, será explorado para avaliar sua eficácia prática e sua capacidade de produzir soluções de qualidade em tempo razoável.

A implementação e análise comparativa desses algoritmos serão fundamentais para compreender as nuances de cada abordagem, permitindo a identificação de cenários específicos nos quais uma estratégia pode se destacar em relação às demais. Ao final, espera-se contribuir não apenas para o avanço teórico no entendimento do TSP, mas também para a aplicação prática desses algoritmos em diferentes contextos, promovendo assim avanços significativos nas soluções para problemas de roteamento e logística.

2. Algoritmo Branch and Bound

O algoritmo Branch and Bound é uma técnica de busca exaustiva utilizada para resolver problemas de otimização combinatória, como o Problema do Caixeiro Viajante (TSP). Ele opera dividindo o espaço de soluções em subconjuntos menores e, em seguida, explorando esses subconjuntos de maneira eficiente para encontrar a solução ótima.



```
Procedure bnb-tsp(A,n)
root = (bound([0]), 0, 0, [0]);
queue = heap([root]);
best = ∞;
sol = ∅;
while queue ≠ ∅ do
    node = queue.pop();
    if node.level > n then
        if best > node.cost then best = node.cost; sol = node.s;
    end
    else if node.bound < best then
        if node.level < n then
            for k = 1 to n − 1 do
                if k ∉ node.s and A[node.s[−1]][k] ≠ ∞ and
                  bound(node.s ∪ {k}) < best then
                    queue.push((bound(node.s ∪ {k}, node.level +
                      1, node.cost + A[node.s[−1]][k], node.s ∪ {k}))
                end
            end
        end
        else if A[node.s[−1]][0] ≠ ∞ and bound(node.s ∪ {0}) < best
          and ∀i ∈ node.s then
            queue.push((bound(node.s ∪ {0}, node.level + 1, node.cost +
              A[node.s[−1]][0], node.s ∪ {0}))
        end
    end
end
```

Figure 1. Pseudo-código para o algoritmo de Branch and Bound [Cormen 2002]

2.1. Escolha da Estimativa de Custo

A escolha da estimativa de custo é crucial para guiar a busca do Branch and Bound. No contexto do TSP, optamos por uma heurística que estima o custo de uma solução parcial. A heurística de inserção mais próxima é comumente escolhida. Em cada etapa do processo, o algoritmo seleciona o nó mais próximo do último visitado para formar uma solução parcial. Isso proporciona uma estimativa eficiente e razoavelmente precisa do custo total da solução.

2.2. Estruturas de Dados Utilizadas

A representação eficiente do grafo é fundamental para o desempenho do algoritmo. No caso do Branch and Bound para o TSP, optamos por utilizar uma matriz de adjacência. Essa escolha se justifica pela facilidade de acesso às distâncias entre as cidades. A matriz de adjacência oferece acesso constante à distância entre quaisquer duas cidades, o que é essencial durante a construção e avaliação de soluções parciais.

2.3. Escolha entre Best-First ou Depth-First

O *Branch and Bound* pode ser implementado de duas formas principais: Best-First e Depth-First. A escolha entre essas estratégias impacta diretamente a eficiência do algoritmo. Optamos pela estratégia Best-First para priorizar a exploração de soluções mais promissoras. Essa abordagem implica na seleção dos nós da árvore de busca com base em suas estimativas de custo, buscando explorar primeiro os ramos que têm maior potencial para levar a soluções ótimas.

2.4. Outros Detalhes de Implementação

Além das escolhas mencionadas, a implementação do Branch and Bound para o TSP requer cuidados adicionais. É necessário rastrear e podar subárvores que não têm potencial para levar a uma solução ótima, reduzindo assim a complexidade computacional. Isso é geralmente feito por meio de limites superiores (upper bounds) e limites inferiores (lower bounds) que são continuamente atualizados durante a busca.

3. Algoritmo de Christofides

O algoritmo de Christofides é uma abordagem polinomial para resolver o Problema do Caixeiro Viajante (TSP), conhecido por fornecer soluções aproximadas eficientes. A seguir, descrevemos os principais elementos da implementação deste algoritmo.

3.1. Estimativa de Custo

O algoritmo de Christofides utiliza uma abordagem mais elaborada para a estimativa de custo. Ele envolve a construção de um circuito euleriano, garantindo que cada aresta seja percorrida exatamente uma vez, e a aplicação de uma técnica de aproximação para obter uma solução que seja no máximo $\frac{3}{2}$ vezes o custo da solução ótima.

3.2. Estruturas de Dados Utilizadas

Mantivemos a representação por matriz de adjacência para garantir a consistência na comparação com o Branch and Bound. Essa escolha facilita a manipulação eficiente das distâncias entre as cidades durante a construção do circuito euleriano.

3.3. Detalhes de Implementação

A implementação do algoritmo de Christofides inclui as seguintes etapas:

1. **Construção de um Minimum Spanning Tree (MST):** Utilizamos um algoritmo eficiente, como Prim ou Kruskal, para obter uma árvore geradora mínima do grafo original.
2. **Criação de um Circuito Euleriano:** Com base no MST, duplicamos as arestas que não pertencem ao MST para formar um circuito euleriano, visitando cada vértice exatamente duas vezes.
3. **Aplicação da Técnica de Aproximação:** Utilizamos uma técnica específica para aproximar o circuito euleriano para obter uma solução do TSP.

4. Algoritmo de Twice Around The Tree

O algoritmo de *Twice Around The Tree* (TAT) é uma técnica que visa encontrar soluções aproximadas eficientes para o Problema do Caixeiro Viajante (TSP). Ele é conhecido por ser uma 2-aproximação para o Problema do Caixeiro Viajante (TSP). Isso significa que a solução encontrada pelo algoritmo tem um custo que é no máximo duas vezes o custo da solução ótima, onde o custo é medido pela soma das distâncias percorridas. A seguir, apresentamos os principais elementos da implementação deste algoritmo.

4.1. Construção do Grafo

O algoritmo inicia pela construção de um grafo completo representando as cidades e suas distâncias. Essa representação é essencial para a execução eficiente do TAT.

4.2. Estratégia de Twice Around The Tree

A abordagem central do TAT envolve o seguinte processo:

1. **Construção de uma Árvore Geradora Mínima (MST):** Utiliza-se um algoritmo eficiente, como Prim ou Kruskal, para obter uma árvore geradora mínima do grafo original.
2. **Duplicação das Arestas:** Com base na MST, duplicam-se as arestas não pertencentes à árvore para formar um circuito euleriano, assegurando que cada vértice seja visitado exatamente duas vezes.
3. **Refinamento da Solução:** Aplica-se técnicas específicas, como a técnica de Twice Around The Tree, para melhorar a solução obtida do circuito euleriano.

5. Comparação entre os algoritmos

5.1. Christofides

O algoritmo de *Christofides* destaca-se ao oferecer soluções aproximadas garantidas para instâncias médias do TSP. Sua implementação eficiente em tempo polinomial o torna uma escolha valiosa em cenários onde a obtenção da solução exata é impraticável devido a restrições de tempo computacional. Se a prioridade é uma solução rápida e aceitável em termos de qualidade, o algoritmo de *Christofides* é uma escolha sólida.

5.2. Twice Around The Tree

O *Twice Around The Tree* (TAT) é uma 2-aproximação para o TSP, proporcionando soluções aproximadas eficientes. Sua capacidade de oferecer resultados com garantias teóricas faz dele uma opção eficaz em cenários onde a solução exata é impraticável devido a restrições de tempo. O TAT destaca-se em fornecer soluções de qualidade aceitável com eficiência computacional.

5.3. Branch and Bound

O *Branch and Bound* (B&B) é um método exato que explora o espaço de soluções de maneira inteligente, evitando ramos que não levariam a uma solução ótima. É especialmente útil para instâncias pequenas do TSP, onde a busca exaustiva é viável. O B&B é a escolha preferencial quando a precisão da solução é crucial e há recursos computacionais disponíveis para explorar todo o espaço de soluções.

5.4. Testes

Nessa secção, vamos comparar os algoritmos aproximativos em relação a qualidade da resposta, tempo de execução e uso da memória (Usei da biblioteca psutil para as comparações de espaço, por isso os valores não são precisos). O algoritmo de Branch and Bound não entrou nas comparações, visto que, nas instâncias mais simples ele já ultrapassou o tempo limite de 30 minutos para execução, ou seja, a visualização dos dados ia ficar prejudicada.

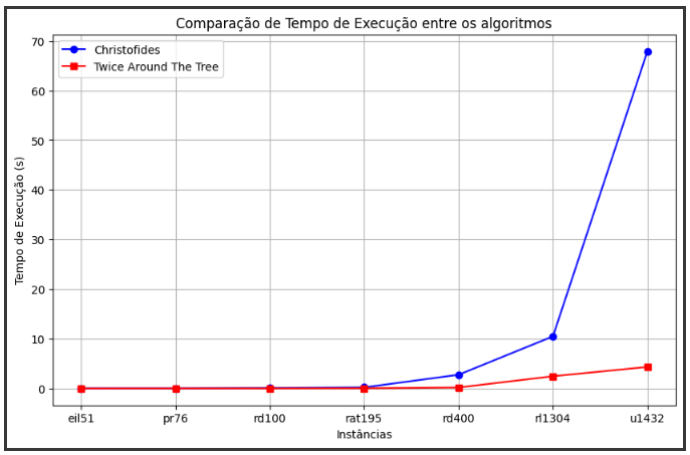


Figure 2. Comparativo entre o tempo de execução dos algoritmos

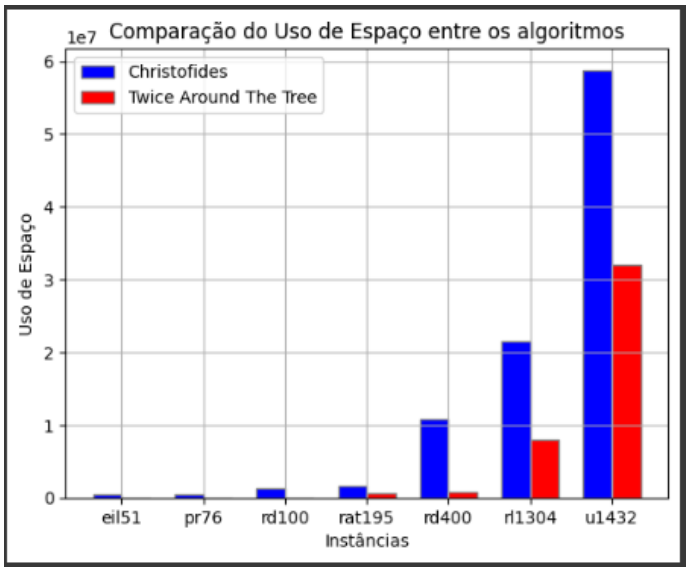


Figure 3. Comparativo entre o espaço utilizado pelos algoritmos

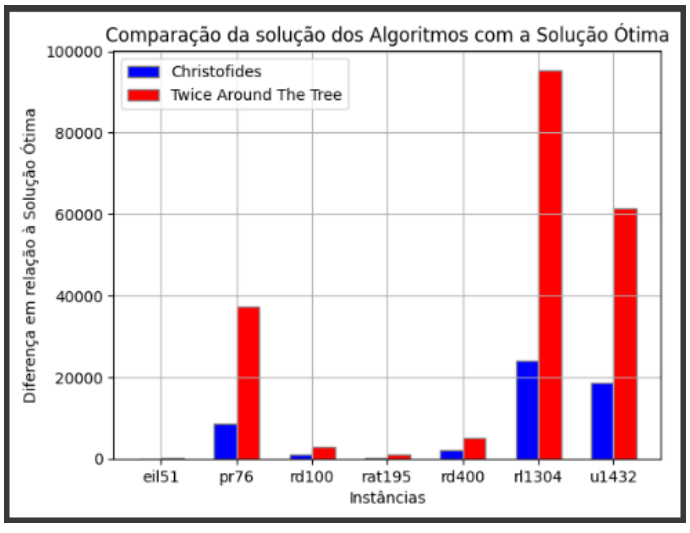


Figure 4. Comparativo entre a qualidade da solução dos algoritmos

6. Conclusão

Em suma, podemos solucionar o problema do caixeiro viajante de diversas maneiras. Em primeiro lugar, o algoritmo exato, por mais que seja implementado com melhorias, ainda possui complexidade exponencial, dessa forma, ele só é viável para casos bem simples. Por outro lado, o algoritmo *Twice Arount The Tree* apresenta uma solução muito veloz porém com uma aproximação de 2, isso significa dizer que nossa resposta pode ser até 2x pior que a ótima, ou seja, o resultado pode não ser tão relevante. Por fim, o algoritmo de *Christofides* fica no meio termo, como possui uma aproximação de 1.5x da solução ótima, seu resultado pode ser mais expressivo, entretanto, o tempo de execução do algoritmo cresce muito mais rapidamente do que o anterior, o que pode tornar casos complexos impossiveis em tempo hábil.

Dessa forma, a escolha entre esses algoritmos dependerá das características específicas do problema em questão, como o tamanho da instância do TSP, a necessidade de uma solução exata e as restrições de tempo computacional. [Levitin 2011]

1. **Para soluções rápidas e aceitáveis:** *Christofides* é preferível.
2. **Para soluções aproximadas eficientes:** *Twice Around The Tree* é uma escolha sólida.
3. **Para soluções exatas em instâncias pequenas:** *Branch and Bound* é a abordagem mais apropriada.

References

Cormen, T. H. (2002). *Algoritmos: Teoria e Prática*. Elsevier Editora Ltda., 6th edition.

Levitin, A. (2011). *Introduction to the Desing Analysis of Algorithms*. Pearson, 3th edition.